CSE574 Introduction to Machine Learning

Dimensionality Reduction

Jue Guo

University at Buffalo

March 29, 2024

Outline

Feature Selection

- Compare and contrast the difference techniques used for dimensionality reduction.
- Use the appropriate statistical test when performing filter-based feature selection techniques.
- Describe the steps in recursive feature elimination.
- Describe the steps in forward and backward sequential feature selection.
- Explain the advantages of using embedded methods over filter-based and wrapper techniques.

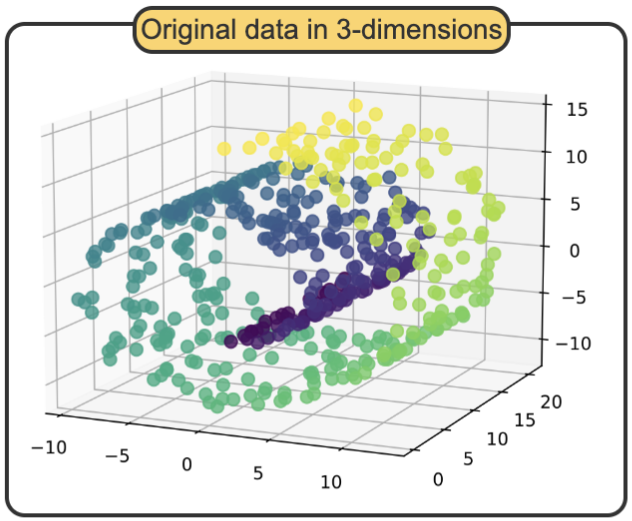
Dimensionality Reduction

Dimensionality Reduction is the process of reducing the number of features in a dataset while preserving as much of the information as possible.

- The number of features can be reduced by combining feature to create new features in a process called *feature extraction* or selecting a subset of features in a process called *feature selection*.
- The primary goal is to simplify the data representation and make data more manageable and interpretable.

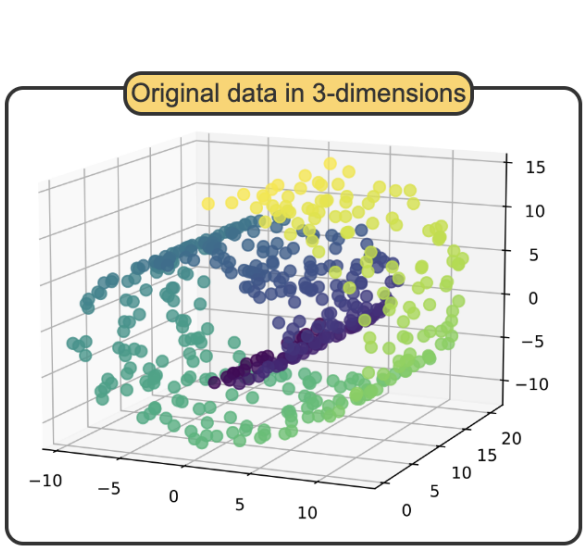
Dimensionality reduction also reduces computation cost, which is very important when working with certain machine learning algorithms.

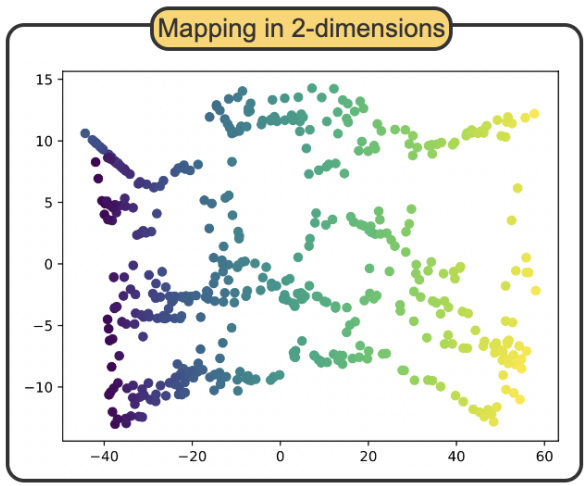
Dimensionality Reduction



Dimensionality reduction is a crucial step in the data-preprocessing pipeline.
Here, a scatter plot of points in 3-dimensions is shown.

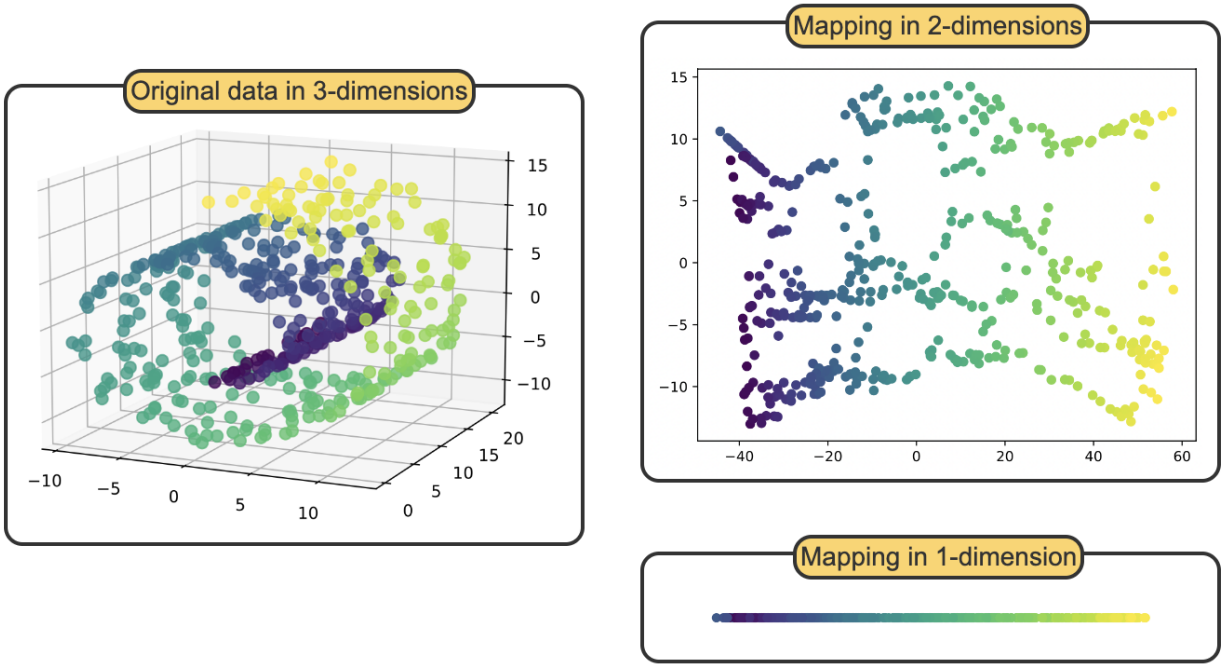
Dimensionality Reduction





Mapping the 3-dimensional data into 2-dimensions can simplify data representation and enhance interpretability.

Dimensionality Reduction



In some cases, data can be reduced to even smaller dimensions without losing too much information.

Introduction to feature selection

Feature selection is a process of selecting a subset of features from a dataset that are most relevant for a machine learning task.

- The main goals of feature selection are to improve model performance and reduce computational complexity.
- Feauture selection methods can be classified into three broad categories: filter-based methods, wrapper methods and embedded methods.

Introduction to feature selection

- **Filter-based methods** rank features based on the relationship between the input and output features.
- **Wrapper methods** select features by iteratively building and evaluating models on a subset of features.
- **Embedded methods** integrate the feature selection processes into the training of a machine learning model.

Filter-based methods

Filter-based methods rank the importance of each feature based on statistical tests involving a single feature. The required statistical test depends on the data type of the input and output features.

- Ex: In the Wisconsin Breast Cancer dataset, the input features are all numeric, while the output feature is categorical, so comparing the F-statistic from a logistic regression model containing a single input feature would be appropriate.

Filter-based methods

Input features	Output feature	
	Numeric	Categorical
Numeric	Pearson correlation coefficient, R	F-statistic
Categorical	F-statistic	χ^2-statistic

Wrapper methods: Recursive feature elimination

A common wrapper method is recursive feature elimination.

- **Recursive feature elimination (RFE)** iteratively removes the least important features from the dataset until a desired number of features is reached.
- The RFE algorithm takes in two hyperparameters : an estimator and the desired number of features.

Select a machine learning model and the desired number of features.

1. Train a machine learning model using all features.
2. Rank the importance of each feature.
3. Eliminate the least important feature.

Repeat steps 1-3 until the desired number of features remains or model performance stabilizes.

Wrapper methods: Sequential feature selection

Since a dataset with p features contains 2^p feature subsets, an exhaustive search for the best performing feature subset is often prohibitive for datasets with a large number of features.

- **Sequential feature selection (SFS)** selects or removes features from the dataset in a step-by-step manner by using cross validation techniques to evaluate model performance at each iteration.

The two main types of SFS are forward selection and backward selection. Although similar, forward and backward selection do not always result in the same model.

The two main types of SFS

- **Sequential forward selection** starts with no features and iteratively adds features to the model starting with the most important feature.
- **Sequential backward selection** starts with all features in the dataset and iteratively removes features from the model starting with the least important feature.

Both forward and backward selection is monotonic, which means that features cannot be removed once added. Similarly, features cannot be added once removed.

Sequential Forward Selection

Suppose a dataset has p features. Begin with an empty feature set. For iteration $i = 1, \ldots, p$:

1. Generate all subsets of size i that contain features from the previous iteration. $p + 1 - i$ subsets are generated at step i. Each subset will contain the chosen subset from the previous iteration and one additional feature.
2. Evaluate model performance on each subset.
3. Keep the added feature from the subset with the best score.

Compare the performance of feature subsets for iterations $1, \ldots, p$. The feature subset with the best evaluation score is selected. For ties, the feature subset with less features should be selected.

Sequential backward selection

Suppose a dataset has p features. Begin with a feature set with all p features. For iteration $i = 1, \ldots, p$:

1. Generate all subsets of size $p - i$ that exclude features removed from the previous iteration. $p + 1 - i$ subsets are generated at step i. Each subset will contain the chosen subset from the previous iteration with one less feature.
2. Evaluate model performance on each subset.
3. Remove the feature that is missing from the subset with the best score.

Compare the performance of feature subsets for iterations $1, \ldots, p$. The feature subset with the best evaluation score is selected. For ties, the feature subset with less features should be selected.

Embedded methods

An embedded method for feature selection selects the most relevant features when training a machine learning model. Embedded methods differ from filter methods and wrapper methods in that feature selection is an integral part of the model building process.

- Embedded methods typically incorporate feature selection as a step within the algorithm itself or through model-specific techniques.
- Common examples are tree-based methods (Ex: random forest) and regularization techniques (Ex: LASSO regression).

Embedded methods

Advantages of embedded methods include:

- Efficiency—Embedded methods are efficient, because feature selection is integrated into the training process. Feature selection is automated and does not involve a separate step.
- Improved interpretability of models—In some cases, embedded methods highlight which features are most influential.
- Flexibility in model selection—Embedded methods select the best features while simultaneously choosing an appropriate machine learning algorithm. This flexibility can be advantageous when exploring various modeling approaches.

Feature extraction using linear techniques

- Perform factor extraction using multiple linear techniques.
- Select the appropriate techniques for different use cases.
- Identify the terminology used in principal component analysis, independent component analysis, and factor analysis and list each algorithm's steps.

Introduction to linear feature extraction

Feature extraction is the process of creating new features from raw data to reduce dimensionality and identify important patterns. Often, raw data cannot be really used for a machine learning algorithm, so data should be transformed into a format from which feature can be extracted.

Feature extraction can lead to improved model performance, reduced training times, and better generalization to new data. **Linear feature extraction** creates new features by taking linear combinations of existing features instead of using all original features. Linear feature extraction techniques include:

- Principal component analysis
- Independent component analysis
- Factor analysis

Principal Component Analysis

Principal Component Analysis, or **PCA**, is a procedure that transforms data into a new coordinate system in which the variance of the data along each axis is maximized.

- **Principal components** are the axes used in PCA. PCA is most commonly used for image compression, noise reduction, and anomaly detection.

An important step in performing PCA is finding the eigendecomposition of the covariance matrix.

- A **covariance matrix** contains the direction of the relationships between features in the dataset.
- **Eigendecomposition** decomposes a covariance matrix into eigenvalues and eigenvectors.

Principal Component Analysis

This decomposition expresses a covariance matrix C as a sequence of scale and rotation transformations. Mathematically, $C = P\Lambda P^T$ where P is a matrix whose columns are eigenvectors, and Λ is a matrix whose diagonals are eigenvalues. In the context of PCA:

- An **eigenvalue** is a scalar that indicates the amount of variance explained by the corresponding principal component.
- An **eigenvector** is a unit vector that describes the direction of the corresponding principal component.

Algorithm: PCA

1. *Standardize the data.* Since PCA is sensitive to scale, data should be standardized to have a mean of 0 and a standard deviation of 1.
2. *Compute the covariance matrix.* The covariance between features X and Y is given by

$$\text{cov}(X, Y) = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{N}$$

where:

- x_i and y_i are the standardized values for the i th instance
- $\bar{x}$ and $\bar{y}$ are the means of the standardized values
- N is the number of instances

The covariance matrix C is a $p \times p$ matrix where p is the number of features and the off-diagonal elements contain the covariances between two features. The diagonal elements contain the variances of each feature.

Algorithm: PCA

3. *Compute the eigenvalues and corresponding eigenvectors.* The eigenvalues and eigenvectors of the covariance matrix C are obtained by solving the equation

$$C\mathbf{v} = \lambda\mathbf{v}$$

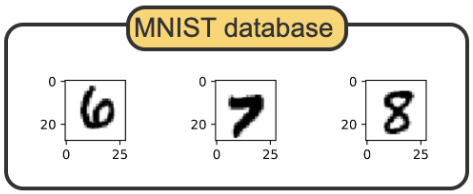
where λ is an eigenvalue and $\mathbf{v}$ is the corresponding eigenvector.

4. *Sort and select the principal components.* The principal components with the largest eigenvalues capture the most variance in the data.
A **scree plot** can be used to choose the number of components to retain. A scree plot is a line plot of explained variance or eigenvalues of principal components. The principal components at or before the elbow or leveling off point should generally be retained.

5. Transform the data. The projection matrix uses the eigenvectors with the highest eigenvalues as columns and transforms the original data into a lower-dimensional space. The projection is obtained using the equation

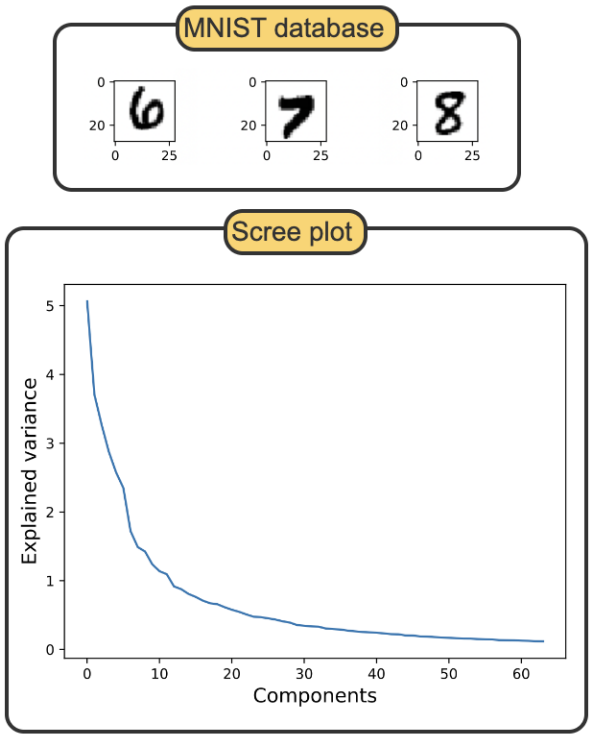
$$\text{new data} = \text{standardized data} * \text{projection matrix}$$

Applying PCA to the MNIST digits dataset.



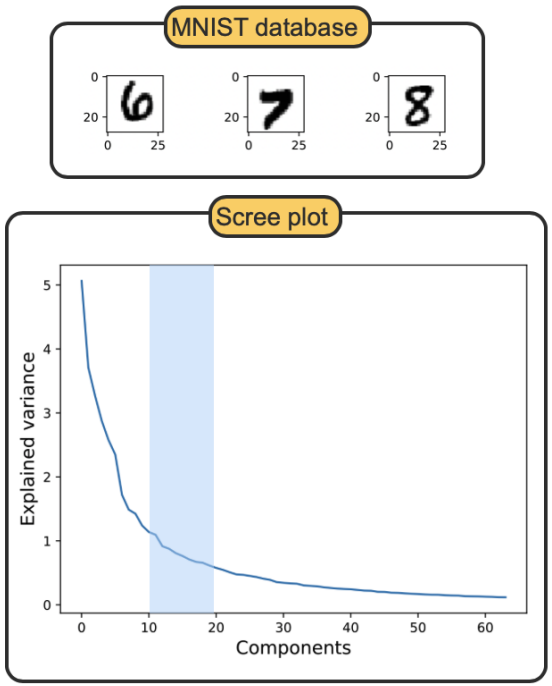
The Modified National Institute of Standards and Technology (MNIST) digits dataset is a collection of handwritten, 28x28 pixel digits commonly used for image processing.

Applying PCA to the MNIST digits dataset.



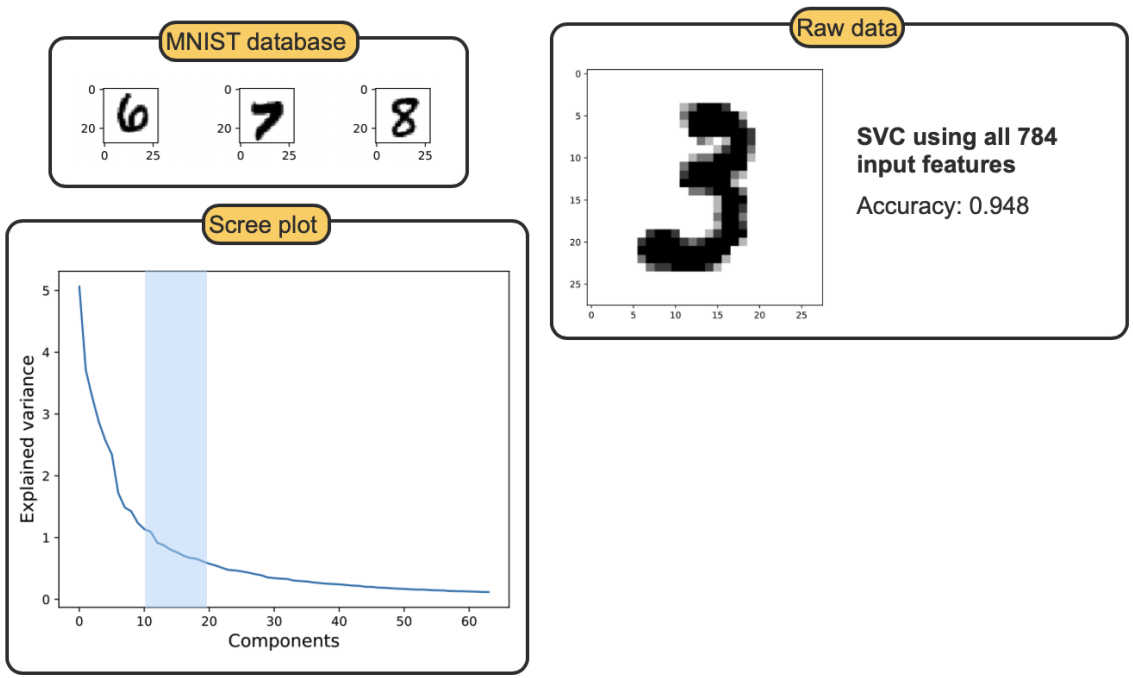
In a PCA model, a scree plot shows a line plot of components vs. explained variance or eigenvalues. A scree plot is helpful in selecting the number of components to be retained.

Applying PCA to the MNIST digits dataset.



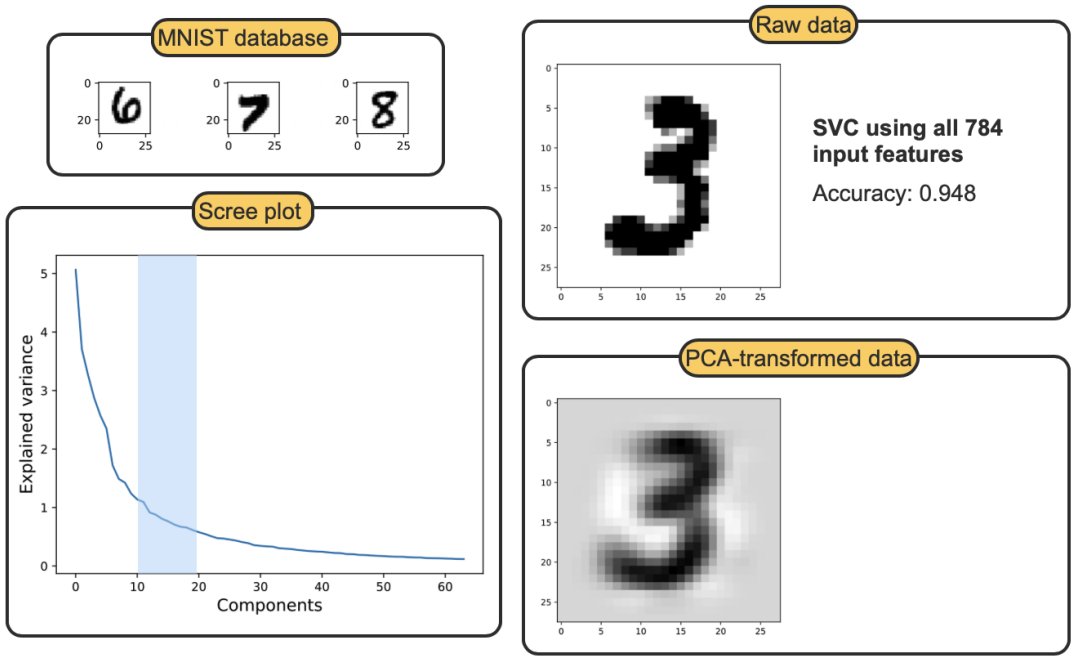
The elbow point is the location in the scree plot where the line plot levels off. The number of components at the elbow point should be retained. Here, the elbow point occurs between 10 and 20 components.

Applying PCA to the MNIST digits dataset.



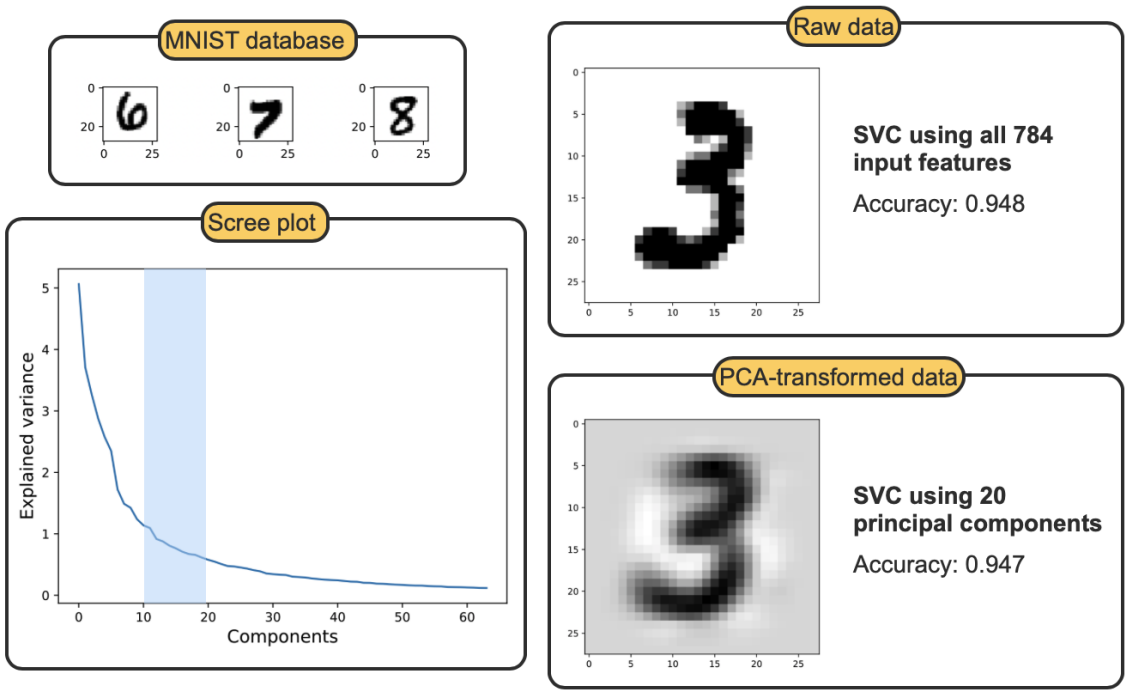
A 28x28 pixel image can be represented using 784 features. Building an SVC model using all 784 features correctly classifies 94.8% of the images in the test set.

Applying PCA to the MNIST digits dataset.



Instead of using the raw images in the training set, the images can be transformed or compressed using PCA before building a classification model.

Applying PCA to the MNIST digits dataset.



Using the principal components results in somewhat lower accuracy but a simpler model overall. An SVC model using 20 principal components yields an accuracy score of 94.7%.